

# Policy Recognition for Multi-Player Tactical Scenarios

Gita Sukthankar  
Robotics Institute  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA  
gitars@cs.cmu.edu

Katia Sycara  
Robotics Institute  
Carnegie Mellon University  
5000 Forbes Ave.  
Pittsburgh, PA  
katia+@cs.cmu.edu

## ABSTRACT

This paper addresses the problem of recognizing policies given logs of battle scenarios from multi-player games. The ability to identify individual and team policies from observations is important for a wide range of applications including automated commentary generation, game coaching, and opponent modeling. We define a *policy* as a preference model over possible actions based on the game state, and a *team policy* as a collection of individual policies along with an assignment of players to policies. This paper explores two promising approaches for policy recognition: (1) a model-based system for combining evidence from observed events using Dempster-Shafer theory, and (2) a data-driven discriminative classifier using support vector machines (SVMs). We evaluate our techniques on logs of real and simulated games played using Open Gaming Foundation d20, the rule system used by many popular tabletop games, including Dungeons and Dragons.

## Categories and Subject Descriptors

I.5 [Pattern Recognition]: Misc.; I.2.1 [Applications and Expert Systems]: Games

## General Terms

Algorithms

## Keywords

policy recognition, multi-player games, Dempster-Shafer evidential reasoning, Support Vector Machines (SVM), plan recognition

## 1. INTRODUCTION

This paper addresses the problem of analyzing multi-player tactical battle scenarios from game logs. The ability to identify individual and team plans from observations is important for a wide range of applications including constructing

opponent models, automated commentary, coaching applications, and surveillance systems. However, the military adage “no plan, no matter how well conceived, survives contact with the enemy intact” reveals that in many cases team plan execution halts early in the course of battle due to unexpected enemy actions. Of course, an ideal plan would include courses of action for all possible contingencies, but typical battle plans only include options for a small set of expected outcomes. If the enemy’s actions cause the world state to deviate from this expected set, the team is often forced to abandon the plan. After multiple plans have been initiated and abandoned, matching the observation trace becomes a difficult proposition, even for an omniscient observer. Moreover, it is unclear whether expert human teams create deep plan trees in situations where enemy actions may force plan abandonment after a few time steps.

Even in cases when the pre-battle plan has been abandoned, we hypothesize that successful teams continue to follow a *policy* through the course of battle and that this policy can be recovered from the observed data. We define a *policy* as a preference model over possible actions, based on the current game state. A *team policy* is a collection of individual policies along with an assignment of players to policies (roles). Policies are typically broad but shallow, covering all possible game states without extending far through time, whereas *plans* are deep recipes for goal completion, extending many time steps into the future, but narrow, lacking contingencies for all but a small set of expected outcomes. The same player intentions can be expressed as either a plan or a policy for game play.

In this paper, we present two techniques for recovering individual and team policies from multi-player tactical scenarios. Our scenarios are described and played using the Open Gaming Foundation d20 Game System (v3.5) [18]. The d20 System is a set of rules governing combat, negotiation, and resource management employed by popular turn-based tabletop games including Dungeons and Dragons and the Star Wars role-playing game.

The remainder of the paper is organized as follows. Section 2 summarizes related work on goal, policy, and plan recognition in games. Section 3 defines the policy recognition problem, describes the d20 rule system, and presents the battle scenarios that are used by our human players. The game logs from these battles provided the data on which our policy recognition approaches are evaluated. Sections 4 and 5 present two complementary methods for policy recognition: an evidential reasoning system for scoring data from game logs and a discriminative classifier trained on simu-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS’07 May 14–18 2007, Honolulu, Hawai’i, USA.  
Copyright 2007 IFAAMAS .

lated game logs. Section 6 discusses results in the context of plan recognition, and Section 7 concludes the paper.

## 2. RELATED WORK

In this section, we present a brief overview of related work on analyzing player actions in games and military scenarios. Single-player keyhole plan recognition was implemented for text-based computer adventure games by Albrecht *et al.* [1] where dynamic Bayesian networks were used to recognize quest goals and to predict future player actions. Mott *et al.* [11] demonstrated a similar goal recognition system for interactive narrative environments using scalable n-gram models. Unlike those systems, our work focuses on the tactical aspect of battlefield adventures where multi-player interactions, limited player knowledge and stochastic action outcomes significantly increase the degree of unpredictability.

Behavior recognition has also been explored in the context of first-person shooter computer games. Moon *et al.* [10] analyzed team effectiveness in America’s Army Game in terms of communication and movement patterns. Rather than recognizing behaviors, their goal was to distinguish between effective and ineffective patterns. Sukthankar and Sycara [16] employed a variant of Hidden Markov Models to analyze military team behaviors in Unreal Tournament based exclusively on the relative physical positioning of agents.

Team behavior recognition in dynamic sports domains has been attempted using both model-based and data-driven approaches. Intille and Bobick [8] developed a framework for recognizing known football plays from multi-agent belief networks that were constructed from temporal structure descriptions of global behavior. Bhandari *et al.* [3] applied unsupervised data mining and knowledge discovery techniques to recognize patterns in NBA basketball data. Recently, Beetz *et al.* [2] developed a system for matching soccer ball motions to different action models using decision-trees. The work on behavior recognition in sports has focused primarily on the mapping of movement traces to low-level game actions (e.g., scoring and passing). By contrast, our paper examines sequences of higher-level agent actions and game state to infer the player’s policy and current tactical role in the team.

Policy recognition has been applied to problems in the Robocup domain. Chernova and Veloso [6] presented a technique to learn an opponent evasion policy from demonstration. Kuhlmann *et al.* [9] fitted a team’s movement patterns to a parametric model of agent behavior for a coaching task. Patterns were scored according to their similarity to models learned from the pre-game logs. This work is conceptually similar to our data-driven approach for policy recognition.

## 3. DOMAIN

To simulate the process of enemy engagement, we adapted the combat section of Open Gaming Foundation’s d20 System (v3.5) to create a set of multi-player tactical scenarios. The d20 System has several useful properties that make it a promising domain:

1. D20 is a turn-based rather than a real-time game system. A game can thus be logged as a sequence of discrete actions performed by each player and policy

recognition can be directly executed on these streams of actions and observed game states.

2. The outcome of actions is stochastic, governed by rolling dice (typically an icosahedral die, abbreviated as *d20*). The rules define the difficulty levels for a broad range of combat tasks; to determine whether a particular action succeeds, the player rolls a d20 and attempts to score a number that is greater than or equal to the difficulty level of the task.
3. Spatial arrangements affect the outcome of many of the combat actions, making the difficulty level easier or harder; for instance, two allies on opposite sides of a target gain a mutual benefit for *flanking* the enemy. Thus, the tactical arrangements of units on the grid can significantly influence the course of a battle. Experienced players coordinate such actions to maximize their chance of success.
4. Teamwork between players is of paramount importance. The opposing forces are designed to be impossible for a single player to defeat. However, certain game rewards are occasionally awarded to the first player to achieve an objective. To succeed, players must simultaneously contribute to team goals in battle while pursuing their own competitive objectives.

A typical d20 Game is played by a group of 3–6 players with one referee. Each player controls one character within the virtual gaming world that the referee brings to life through verbal descriptions and miniatures on a dry-erase tabletop gaming map (Figure 1). Characters have a well-defined set of capabilities that determine their competence at various tasks in the virtual world. The referee controls the actions of all other entities in the virtual world (known as *non-player characters*). During a typical four hour session, the referee poses a series of challenges—diplomatic negotiations, battles, and puzzles—that the players must cooperatively solve. Success for the players enhances the capabilities of their characters, giving them more abilities and resources. Failure can result in the death of characters or the loss of resources. Thus, characters’ capabilities persist over multiple gaming sessions which makes it different from other iterative games where state is reset at the conclusion of each session.

Although tabletop gaming lacks the audio and visual special effects of its increasingly-popular computerized cousins, tabletop games have a stronger emphasis on team tactics and character optimization. In battle, time becomes a limited resource and players typically have to coordinate to overcome their foes before the foes defeat them. Since most computer games are real-time rather than turn-based, they disproportionately reward fast keyboard reflexes and manual dexterity over tactical and strategic battle planning.

### 3.1 Multi-Player Tactical Scenarios

Our experiments in policy recognition focus on the subset of the d20 system actions that deal with tactical battles (as opposed to diplomatic negotiation or interpersonal interaction). Each scenario features a single battle between a group of player characters and a set of referee-controlled opponents. The players have limited knowledge of the world state and limited knowledge about their opponents’ capabilities and current status; the referee has complete knowledge of the entire game state. Players are allowed to communicate in order to facilitate coordination but the referee will



Figure 1: A tactical battle in the d20 System. Players control characters represented by plastic miniatures in a discretized grid environment. Actions with stochastic outcomes are resolved using dice, with probabilities of a successful outcome listed in rulebooks. A human referee (not shown) adjudicates the legality of player actions and controls opponents, such as the dragon.

Table 1: Characters and summarized capabilities

Name	Offensive	Defensive	Magic	Stealth
A	high	medium	low	low
B	medium	high	low	low
C	low	medium	medium	low
D	high	low	medium	medium
E	medium	low	low	high
F	medium	low	high	medium

usually disallow excessive communication during battle and limit players to short verbal utterances.

Figure 2 shows a typical multi-player tactical scenario from Dungeons & Dragons. The three players must cooperate to achieve one of two objectives: defeat the dragon, or distract the dragon to rescue the prisoner in the corner of the room. Based on the capabilities of their characters, the players select a goal and allocate functional roles for the upcoming battle. Each of the three players was assigned a character originally developed for the 2006 Dungeons and Dragons Open Championships [7]; their capabilities are summarized in Table 1. Each character is capable of fulfilling multiple roles in a team but each is poorly suited to at least one role. The roles are:

1. **slayer** who fights the dragon in close-combat;
2. **blocker**, a defensive character who shields more vulnerable characters;
3. **sniper**, who skirmishes the dragon at range;
4. **medic** who restores health to other characters;
5. **scout**, a stealthy character who can rescue the prisoner without being noticed by the distracted dragon.

### 3.2 Game Mechanics

The battle sequence in each scenario is as follows:

1. The referee draws the terrain and places the opponents on the grid at the beginning of the scenario. Then the

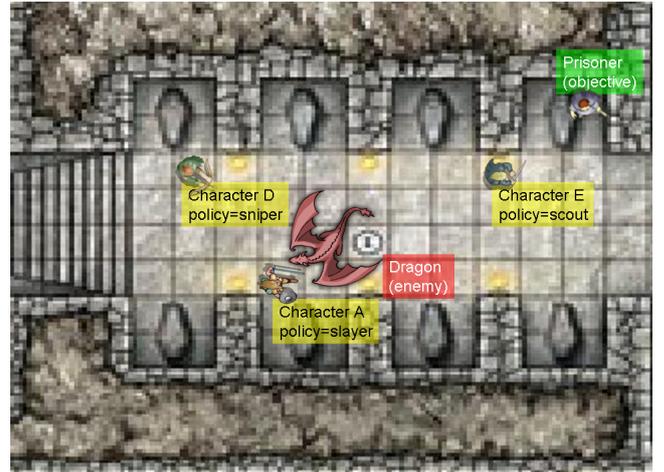


Figure 2: Primary tactical scenario. The three human players select roles that enable them to achieve one of two goals: (1) defeat the dragon in battle; (2) distract the dragon while one character frees the prisoner.

players place their miniatures on the grid to indicate their starting location.

2. At the start of the battle, each entity (players and opponents) rolls an initiative to determine the order of action each round (time unit). This initiative is maintained through the battle (with a few possible exceptions).
3. Each round is a complete cycle through the initiatives in which each entity can move and take an action from a set of standard actions.
4. If a character's health total goes below 0, it is dead or dying and can no longer take any actions in the battle unless revived by another player's actions.

Each character has the following attributes that affect combat actions: (1) **hit points**: current health total; (2) **armor class**: a measure of how difficult a character is to hit with physical attacks; (3) **attack bonus**: a measure of how capable a character is at handling weaponry. When a character attacks an opponent, the attack is resolved using the following formula to determine whether the attack is successful:

$$d(20) + \text{attack bonus} + \text{modifiers} \geq \text{armor class} + \text{modifiers}$$

where  $d(20)$  is the outcome of a twenty-sided die roll. If the expression is true, the attack succeeds and the opponent's hit points are reduced.<sup>1</sup> Situational modifiers include the effect of the spatial positioning of the characters. For instance, the defender's armor class improves if the attacker is partially occluded by an object that provides cover.

### 3.3 Problem Formulation

We define the problem of policy recognition in this domain as follows. Given a sequence of input observations  $\mathcal{O}$  (including observable game state and player actions) and a

<sup>1</sup>For Dungeons and Dragons, which is based in a fantasy setting, a similar system details how different characters can employ magic to attack opponents and how resistant characters are to the effects of magic.

set of player policies  $\mathcal{P}$  and team policies  $\mathcal{T}$ , the goal is to identify the policies  $\rho \in \mathcal{P}$  and  $\tau \in \mathcal{T}$  that were employed during the scenario. A player policy is an individual’s preference model for available actions given a particular game state. For instance, in the scenario shown in Figure 1, the archer’s policy might be to preferentially shoot the dragon from a distance rather than engaging in close combat with a sword, but he/she might do the latter to protect a fallen teammate. In a team situation, these individual policies can be used to describe a player’s role in the team (e.g., combat medic). A team policy is an allocation of players to these tactical roles and is typically arranged prior to the scenario as a locker-room agreement [15]. However, circumstances during the battle (such as the elimination of a teammate or unexpected enemy reinforcements) can frequently force players to take actions that were *a priori* lower in their individual preference model.

In particular, one difference between policy recognition in a tactical battle and typical plan recognition is that agents rarely have the luxury of performing a pre-planned series of actions in the face of enemy threat. This means that methods that rely on temporal structure, such as Dynamic Bayesian Networks (DBNs) and Hidden Markov Models are not necessarily well-suited to this task. An additional challenge is that, over the course of a single scenario, one only observes a small fraction of the possible game states, which makes policy learning difficult. Similarly, some games involve situations where the goal has failed and the most common actions for a policy are in fact rarely observed (e.g., an enemy creates a smokescreen early in the battle, forcing the archer to pursue lower-ranked options). The following sections present two complementary approaches to policy recognition: (1) a model-based method for combining evidence from observed events using Dempster-Shafer theory, and (2) a data-driven discriminative classifier using support vector machines (SVMs).

## 4. MODEL-BASED POLICY RECOGNITION

The model-based method assigns evidence from observed game events to sets of hypothesized policies. These beliefs are aggregated using the Dempster-Shafer theory of evidential reasoning [14]. The primary benefit of this approach is that the model generalizes easily to different initial starting states (scenario goals, agent capabilities, number and composition of the team).

### 4.1 Dempster-Shafer Theory

This section presents a brief overview of the Dempster-Shafer theory of evidential reasoning [14]. Unlike traditional probability theory where evidence is associated with mutually-exclusive outcomes, the Dempster-Shafer theory quantifies belief over *sets* of events. The three key notions of Dempster-Shafer theory are: (1) basic probability assignment functions ( $m$ ); (2) belief functions (**Bel**); (3) plausibility functions (**Pl**). We describe these below.

The basic probability assignment function assigns a number between 0 and 1 to every combination of outcomes (the power set). Intuitively this represents the belief allocated to this subset and to no smaller subset. For example, after observing an agent’s actions over some time, one may assert that it is following either policy  $\rho_1$  or  $\rho_2$ , without further committing belief as to which of the two is more likely. This

contrasts with the standard Bayesian approach that would typically impose a symmetric, non-informative prior over  $\rho_1$  and  $\rho_2$  (asserting that they were equally likely). More formally, given a finite set of outcomes  $\Theta$  whose power set is denoted by  $2^\Theta$ , the basic probability assignment function,  $m : 2^\Theta \mapsto [0, 1]$  satisfies:

$$\begin{aligned} m(\emptyset) &= 0 \\ \sum_{A \subseteq \Theta} m(A) &= 1. \end{aligned}$$

Following Shafer [14], the quantity  $m(A)$  measures the belief committed *exactly* to the subset  $A$ , not the total belief committed to  $A$ . To obtain the measure of the total belief committed to  $A$ , one must also include the belief assigned to all proper subsets of  $A$ . Thus, we define the belief function  $\text{Bel} : 2^\Theta \mapsto [0, 1]$  as

$$\text{Bel}(A) = \sum_{B \subseteq A} m(B).$$

Intuitively the belief function quantifies the evidence that directly supports a subset of outcomes. The non-informative belief function (initial state for our system) is obtained by setting:  $m(\Theta) = 1$  and  $m(A) = 0 \quad \forall A \neq \Theta$ .

The plausibility function quantifies the evidence that does not directly contradict the outcomes of interest. We define the plausibility function  $\text{Pl} : 2^\Theta \mapsto [0, 1]$  as

$$\text{Pl}(A) = \sum_{B \cap A \neq \emptyset} m(B).$$

The precise probability of an event is lower-bounded by its belief and upper-bounded by its plausibility functions, respectively.

We employ Dempster-Shafer theory to model how observed evidence affects our beliefs about a character’s current policy. For instance, seeing a character moving on the battlefield could indicate that the agent’s role is that of a **sniper**, a **medic** or a **scout** (rather than a **slayer** or **blocker**). This can be expressed as:

$$m(\{\text{sniper}, \text{medic}, \text{scout}\}) = 0.7 \text{ and } m(\Theta) = 0.3.$$

The **belief** that the agent is adopting one of these roles is 0.7, yet the belief that the agent is specifically a sniper is 0 (although the **plausibility** for either of these is 1). Conversely, while the **belief** that the agent is adopting a slayer policy is also 0, the plausibility is only 0.3.

Dempster-Shafer theory also prescribes how multiple, independent sources of evidence should be combined. Dempster’s rule of combination [14] is a generalization of Bayes’ rule and aggregates two basic probability assignments  $m_1$  and  $m_2$  using the following formula:

$$\begin{aligned} m_{12}(\emptyset) &= 0 \\ m_{12}(C \neq \emptyset) &= \frac{\sum_{A \cap B = C} m_1(A)m_2(B)}{1 - \sum_{A \cap B = \emptyset} m_1(A)m_2(B)}. \end{aligned}$$

One potential issue with this rule is its treatment of conflicting evidence. The normalizing term in the denominator redistributes the probability mass associated with conflicting evidence among the surviving hypotheses. Under certain conditions, this has the undesirable property of generating counterintuitive beliefs. In the pathological case, an outcome judged as unlikely by both  $m_1$  and  $m_2$  can have a

value of 1 in  $m_{12}$  if all other subsets conflict. To address this problem, several other rules of combination have been proposed [13].

Yager’s rule [19] is very similar to Dempster’s rule with the single exception that conflicting evidence is assigned to the universal set,  $m(\Theta)$ , rather than used as a normalizer. The rule can be stated as follows:

$$\begin{aligned} m_{12}(\emptyset) &= 0 \\ m_{12}(C \neq \emptyset) &= \sum_{A \cap B = C} m_1(A)m_2(B) \\ m_{12}(\Theta) &\leftarrow m_{12}(\Theta) + \sum_{A \cap B = \emptyset} m_1(A)m_2(B). \end{aligned}$$

Although this formulation is not associative, we implement Yager’s rule in a quasi-associative manner to enable efficient online update [13]. In the absence of conflict, Yager’s rule gives the same results as Dempster’s rule of combination.

Finally, we consider another quasi-associative rule: the intuitive idea of averaging corresponding basic probability assignments [13]:

$$m_{1\dots n}(A) = \frac{1}{n} \sum_{i=1}^n m_i(A).$$

Unfortunately it is impossible to know *a priori* which rule will perform well in a given domain since no single rule has all desirable properties.

## 4.2 Empirical Evaluation

Based on domain knowledge, we identified a general set of observable events that occur during a Dungeons and Dragons battle. Each of these events was associated with a basic probability assignment function to assign beliefs over sets of individual policies. For example, the observed event of a character being attacked by an opponent is associated with:  $m(\Theta) = 0.1, m(\{\text{scout}\}) = 0.4, m(\{\text{blocker}\}) = 0.5$ . This rule assigns a large belief (0.5) to the blocker policy, while reducing the plausibility of the scout policy to 0.1. Note that the set  $\{\text{scout}\}$  also includes the **blocker** policy, thus the plausibility of **blocker** is 1. The plausibility of the remaining three policies is 0.5.

Data from a series of Dungeons and Dragons games using the tactical scenario shown in Figure 2 was recorded and annotated according to our list of observable events. The  $m$ -functions for the set of events observed for each character was aggregated using the three rules of combination described in Section 4.1.

We computed the average accuracy over the set of battles for each of the three rules of combination. At the conclusion of each battle, the system made a forced choice, for each player, among the set of policies (roles). Each player was classified into the singleton policy with the highest belief. Comparing this against the ground truth and averaging over battles produces the confusion matrix given in Table 2. We note that, according to this forced-choice metric, all of the combination rules perform reasonably well, with Dempster’s Rule scoring the best. The largest source of confusion is that the **slayer** policy is occasionally misclassified as **blocker**. This motivates the data-driven method described in Section 5.1 where we specifically learn classifiers to discriminate between these two similar policies.<sup>2</sup>

<sup>2</sup>The **blocker** and **slayer** policies can generate very simi-

To illustrate how belief changes as evidence is aggregated using the different combination rules, we plot the belief for each policy for one battle from our dataset (Figure 3). Since neither Yager nor averaging employ normalization, we plot their beliefs on a semi-log scale. We note the following. Yager’s rule makes conflicting evidence explicit by allocating significant mass to the **unknown** policy. In particular, this reveals the difficulty of distinguishing between the **blocker** and **slayer**, even late in the battle. A concern with Yager’s rule is that the belief for a policy decays over time, despite increasing evidence because all of the rules leak some mass to the **unknown** set. Averaging corresponding  $m$ -values performs the least well in our domain.

## 5. DATA-DRIVEN POLICY RECOGNITION

To discriminate between similar policies, we propose a data-driven classification method that is trained using simulated battle data. By training the method on a specific scenario, it can exploit subtle statistical differences between the observed outcomes of similar policies. For instance, characters following the **slayer** policy should both inflict more damage on their opponents and receive more damage in return, whereas the more defensive **blocker** policy occupies the enemy without resulting in substantial losses on either side. This section describes the classifier that we employ, Support Vector Machines (SVM), and evaluates the method on a second scenario (Figure 4).

### 5.1 Support Vector Machines

The goal of policy classification is to label an observed action sequence as a member of one of  $k$  categories (e.g., **blocker** vs. **slayer**). We perform this classification using support vector machines [17]. Support vector machines (SVM) are a supervised binary classification algorithm that have been demonstrated to perform well on a variety of pattern classification tasks. Intuitively the support vector machine projects data points into a higher dimensional space, specified by a kernel function, and computes a maximum-margin hyperplane decision surface that separates the two classes. Support vectors are those data points that lie closest to this decision surface; if these data points were removed from the training data, the decision surface would change. Given a labeled training set  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_l, y_l)\}$ , where  $\mathbf{x}_i \in \mathbb{R}^N$  is a feature vector and  $y_i \in \{-1, +1\}$  is its binary class label, an SVM requires solving the following optimization problem:

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^l \xi_i$$

constrained by:

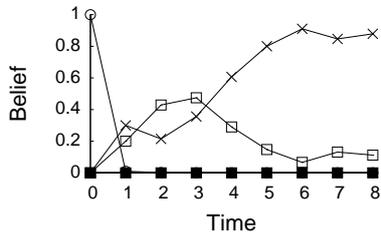
$$\begin{aligned} y_i(\mathbf{w}^T \phi(\mathbf{x}_i) + b) &\geq 1 - \xi_i, \\ \xi_i &\geq 0. \end{aligned}$$

The function  $\phi(\cdot)$  that maps data points into the higher dimensional space is not explicitly represented; rather, a *kernel* function,  $K(\mathbf{x}_i, \mathbf{x}_j) \equiv \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$ , is used to implicitly specify this mapping. In our application, we use the popular

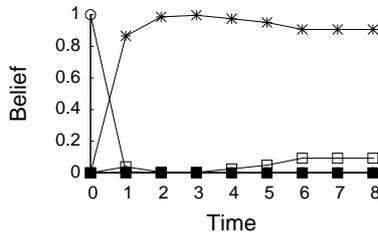
lar observable state since intelligent opponents often target the **slayer** preferentially in order to eliminate their biggest threat.

**Table 2: Confusion matrix for model-based policy recognition**

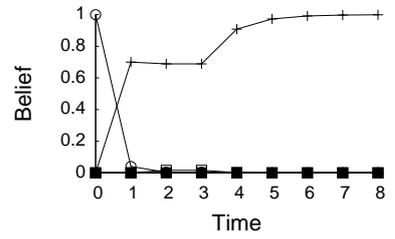
	Dempster					Yager					Averaging				
	sniper	slayer	medic	blocker	scout	sniper	slayer	medic	blocker	scout	sniper	slayer	medic	blocker	scout
sniper	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	88%	0%	0%	12%	0%
slayer	0%	83%	0%	17%	0%	0%	83%	0%	17%	0%	7%	40%	7%	39%	7%
medic	0%	0%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	0%	100%	0%
blocker	0%	0%	0%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	100%	0%
scout	0%	0%	0%	0%	100%	0%	0%	0%	0%	100%	0%	0%	0%	0%	100%



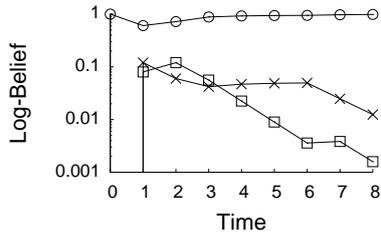
(a) Character A, policy=slayer



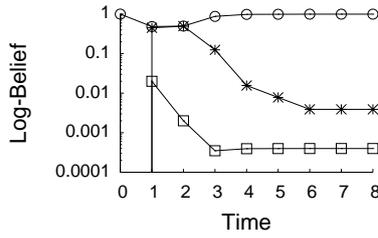
(b) Character C, policy=medic



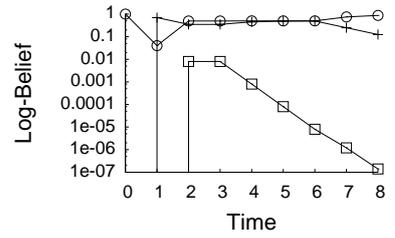
(c) Character E, policy=sniper



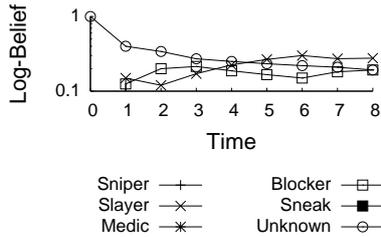
(d) Character A, policy=slayer



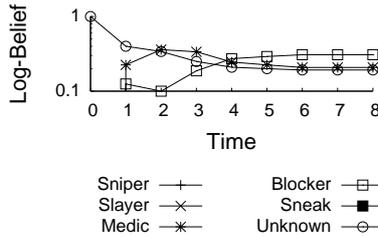
(e) Character C, policy=medic



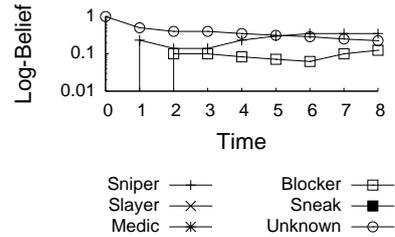
(f) Character E, policy=sniper



(g) Character A, policy=slayer



(h) Character C, policy=medic



(i) Character E, policy=sniper

**Figure 3: Evolution of beliefs over the course of one battle from the scenario shown in Figure 2. The beliefs for policies adopted by each of the three characters, according to the three rules of combination, are shown. The three rows correspond to Dempster's Rule, Yager's Rule and  $m$ -function averaging, respectively. The columns correspond to three characters A, C, E from Table 1, adopting the ground-truth policies, slayer, medic and sniper, respectively.**

**Table 3: Confusion matrix for data-driven policy recognition on the scenario shown in Figure 4 using the three different feature sets.**

	action		state		combined	
	blocker	slayer	blocker	slayer	blocker	slayer
blocker	96%	4%	82%	18%	98%	2%
slayer	0%	100%	16%	84%	0%	100%

radial basis function (RBF) kernel:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0.$$

Many efficient implementations of SVMs are publicly available; we use LIBSVM [5] because it includes good routines for automatic data scaling and model selection (appropriate choice of  $C$  and  $\gamma$  using cross-validation). To use SVMs for  $k$ -class classification, we train  ${}^k C_2$  pair-wise binary classifiers and assign the most popular label.

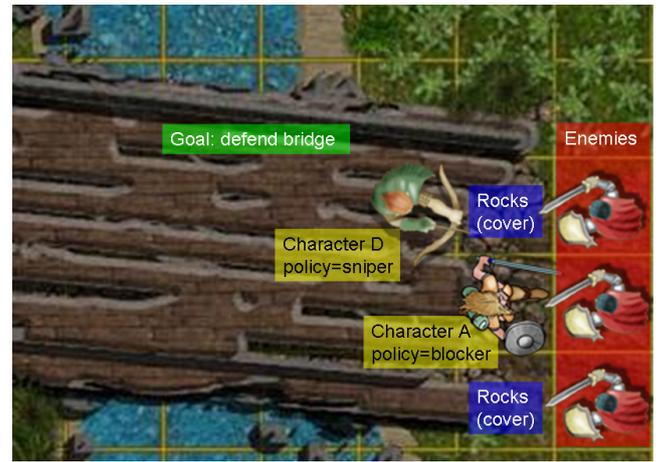
## 5.2 Empirical Evaluation

The data-driven method takes as input a feature vector summarizing the observed information about the battle and performs a forced classification into policies. We investigated three choices for feature sets: (1) a histogram of the observed character actions over the battle — this is similar to a bag of words model in information retrieval; (2) a vector with character and enemy status at every time step; (3) a concatenation of these two vectors.

To train the SVM, we generated training data by simulating a set of single player battles in a simplified scenario, using the policies of interest (**blocker** and **slayer**). The trained SVM was then evaluated on several other scenarios. Table 3 shows the confusion matrices for battles on one of these scenarios. This scenario, shown in Figure 4, is a two-player battle where the characters defend a bridge against multiple opponents; the goal is to correctly classify the policy employed by the front-line character. We observe that the data-driven method using the combined set of features can reliably discriminate between the **blocker** and **slayer** policies. The other two matrices indicate that, in this case, the classifier relies mainly on the histogram of observed actions. However, we note that such data-driven methods require sufficient quantities of training data to avoid overfitting, and that they generalize poorly to novel scenarios when such data has not been provided.

## 6. DISCUSSION

One interesting aspect about battles in the d20 System is that a player’s action choices are constrained by a complex interaction between the character’s capabilities, its location on the map and its equipment (including consumable resources). Hence, different players have different attack options, and each player has different choices over time, depending on the current state of the battle. Despite the stochastic nature of the domain, players typically follow battle tactics that are identifiable to other humans. As the human referee controlling the opponents recognizes the players’ tactics, he/she will often intelligently adapt to the players’ tactics better than computer game engines that are relatively insensitive to player actions. An application of our work would be the development of computerized opponents that react realistically to player tactics to enhance both computer



**Figure 4: Tactical scenario where two players defend a bridge against multiple opponents.**

games and military simulations. Despite its fantasy setting with dragons and magic, the d20 tabletop system exercises many of the tactical concepts that current military commanders employ in decision-making games and situational awareness exercises [12].

Teamwork is an important aspect of tactics in the d20 system since the actions of other players can significantly affect the difficulty level of various combat actions. Although building a fully-specified team plan is typically impractical given the complexity of the domain, players generally enter battle with a “locker-room agreement” specifying the policy that the character will seek to use in the upcoming confrontation. There is not a simple mapping between a character capabilities and this policy; an effective team role must consider the capabilities of team members, the expected abilities of opponents and the overall team goal. We believe that identifying each character’s policy is an important first step towards predicting the character’s future actions, identifying the set of team goals, and generating an automated higher-level commentary on the scenario.

The model-based and data-driven approaches are very complementary approaches to battle analysis. The model-based approach generalizes well to other sets of characters, different opponent types, and variations in scenario. The data-driven classifier is able to detect subtle statistical differences in action and game state sequences to correctly classify externally-similar policies. The data-driven approach does not generalize as well to different character capabilities since a character’s capabilities are implicitly incorporated into the training set; thus a testing set that is statistically-different cannot be classified accurately. We believe that the two approaches should be combined into a hybrid system where the model-based recognizer identifies high-belief policy sets and the data-driven classifier discriminates between those specific policies. Such an approach is similar in spirit to Carberry’s work on incorporating Dempster-Shafer beliefs to focus heuristics for plan recognition [4].

## 7. CONCLUSION

This paper explores two promising approaches for policy recognition: (1) a model-based system for combining

evidence from observed events using Dempster-Shafer theory, and (2) a data-driven classification using support vector machines (SVMs). Evaluation of our techniques on logs of real and simulated games demonstrate that we can recognize player policies with a high degree of accuracy.

Using our game logging methodology and domain-generated  $m$ -functions, the model-based approach performs extremely well over a broad range of initial conditions. Dempster's rule slightly outperforms the other two rules on the forced classification task. The majority of errors involve confusion between the **blocker** and **slayer** policies, which appear similar at a coarse level. To address this issues, we trained a set of discriminative classifiers using simulated battle logs and evaluated the effects of different feature vectors. The resulting classifiers are highly accurate at classifying these policies, although they do not generalize to characters' with different capabilities. Thus, our two approaches are complementary and could be combined into a hybrid policy recognition system to provide detailed automated battle commentary of multi-player tactical scenarios.

## 8. ACKNOWLEDGEMENTS

The authors would like to thank Rahul Sukthankar for his research suggestions. This work has been supported by AFOSR grant F49620-01-1-0542 and is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001.

## 9. REFERENCES

- [1] A. Albrecht, I. Zukerman, and A. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *Journal of User Modeling and User-Adapted Interaction*, 9:5–47, 1998.
- [2] M. Beetz, J. Bandouch, S. Gedili, N. v. Hoyningen-Huene, B. Kirchlechner, and A. Maldonado. Camera-based observation of football games for analyzing multi-agent activities. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems*, 2006.
- [3] I. Bhandari, E. Colet, J. Parker, Z. Pines, R. Pratap, and K. Ramanujam. Advanced Scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, 1(1):121–125, 1997.
- [4] S. Carberry. Incorporating default inferences into plan recognition. In *Proceedings of National Conference on Artificial Intelligence*, 1990.
- [5] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [6] S. Chernova and M. Veloso. Tree-based policy learning in continuous domains through teaching by demonstration. In *Proceedings of Workshop on Modeling Others from Observations (MOO 2006)*, 2006.
- [7] D&D Open Characters, 2006. <http://www.wizards.com/default.asp?x=rpga/conventions/genconindy-opencharacters>.
- [8] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual evidence. In *Proceedings of National Conference on Artificial Intelligence*, 1999.
- [9] G. Kuhlmann, W. Knox, and P. Stone. Know thine enemy: A champion RoboCup coach agent. In *Proceedings of National Conference on Artificial Intelligence*, 2006.
- [10] I.-C. Moon, K. Carley, M. Schneider, and O. Shigiltchhoff. Detailed analysis of team movement and communication affecting team performance in the America's Army Game. Technical Report CMU-ISRI-TR-04-100, Carnegie Mellon University, 2005.
- [11] B. Mott, S. Lee, and J. Lester. Probabilistic goal recognition in interactive narrative environments. In *Proceedings of National Conference on Artificial Intelligence*, 2006.
- [12] J. Phillips, M. McCloskey, P. McDermott, S. Wiggins, and D. Battaglia. Decision-centered MOUT training for small unit leaders. Technical Report 1776, U.S. Army Research Institute for Behavioral and Social Sciences, 2001.
- [13] K. Sentz and S. Ferson. Combination of evidence in Dempster-Shafer theory. Technical Report SAND2002-0835, Sandia National Labs, 2002.
- [14] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [15] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 12:pp.241–273, 1999.
- [16] G. Sukthankar and K. Sycara. Robust recognition of physical team behaviors using spatio-temporal models. In *Proceedings of International Conference on Autonomous Agents and Multi-agent Systems*, 2006.
- [17] V. Vapnik. *Statistical Learning Theory*. Wiley & Sons, Inc, 1998.
- [18] d20 v3.5 System Reference Document, 2003. <http://www.wizards.com/default.asp?x=d20/article/srd35>.
- [19] R. Yager. On the Dempster-Shafer framework and new combination rules. *Information Sciences*, 41:pp.93–137, 2000.